1 The BICGSTAB Algorithm: Setup

The biconjugate gradient stabilized method combines ideas of both CGS and SOR. Particular, we look for an algorithm such that the residuals and the search directions are given by:

\[ r_j = \psi_j(A)\phi_j(A)r_0, \]
\[ p_j = \psi_j(A)\pi_j(A)r_0. \]

Here \( \phi_j \) and \( \pi_j \) are the same polynomials as before in CGS, and they give rise to the residual and the search direction of the original biconjugate gradient algorithm.

The rest polynomial \( \psi_j(t) \) is defined recursively by:

\[ \psi_{j+1}(t) = (1 - \omega_j t) \psi_j(t), \]

in which \( \omega_j \) is the “stabilizing” parameter to be determined later. Recall that \( \phi_j \) and \( \pi_j \) satisfy the recursive relation:

\[ \phi_{j+1}(t) = \phi_j(t) - \alpha_j t \pi_j(t), \]
\[ \pi_{j+1}(t) = \phi_j(t) + \beta_j \pi_j(t); \]

then the update formula for \( r_j \) is:

\[ r_{j+1} = (I - \omega_j A)(r_j - \alpha_j A p_j). \]

The formula for \( p_{j+1} \) can be determined from \( r_{j+1} \):

\[ p_{j+1} = r_{j+1} + \beta_j (I - \omega_j A) p_j. \]

We would like to compute the two scalars \( \alpha_j \) and \( \beta_j \). For example, \( \beta_j = \frac{\rho_{j+1}}{\rho_j} \) where:

\[ \rho_j = (\phi_j(A)r_0) \cdot (\phi_j(A^t)r'_0) = (\phi_j^2(A)r_0) \cdot r'_0. \]

Unfortunately this quantity is not computable from \( r_j \) and \( p_j \). What we can compute is:

\[ \tilde{\rho}_j = (\phi_j(A)r_0) \cdot (\psi_j(A^t)r'_0) = (r_j, r'_0). \]

In order to relate \( \rho_j \) and \( \tilde{\rho}_j \), we write:

\[ \psi_j(t) = a_0^{(j)} t^j + a_1^{(j)} t^{j-1} + \cdots + a_{j-1}^{(j)} t + a_j^{(j)}, \]
\[ \phi_j(t) = b_0^{(j)} t^j + b_1^{(j)} t^{j-1} + \cdots + b_{j-1}^{(j)} t + b_j^{(j)}. \]
From the BCG procedure, \( \phi_j(A)r_0 \) is orthogonal to \( \phi_k(A^t)r'_0 \) for all \( k < j \), see (3.5a) from previous lecture; hence \( (\phi_j(A)r_0) \cdot ((A^t)^j - k r'_0) = 0 \) for all \( k > 1 \), i.e.

\[
\tilde{\rho}_j = \sum_{k=0}^{j} (\phi_j(A)r_0) \cdot (a_k^{(j)}(A^t)^j - k r'_0) = a_0^{(j)}(\phi_j(A)r_0) \cdot ((A^t)^j r'_0),
\]

\[
\rho_j = \sum_{k=0}^{j} (\phi_j(A)r_0) \cdot (b_k^{(j)}(A^t)^j - k r'_0) = b_0^{(j)}(\phi_j(A)r_0) \cdot ((A^t)^j r'_0),
\]

or we derive:

\[
\tilde{\rho}_j = \frac{a_0^{(j)}}{b_0^{(j)}} \rho_j. \tag{1.12}
\]

Following [1.3] we have:

\[
a_0^{(j+1)} = -\omega_j a_0^{(j)}, \tag{1.13}
\]

and following [1.6] and [1.7] we have:

\[
b_0^{(j+1)} = -\alpha_j b_0^{(j)}. \tag{1.14}
\]

As a result:

\[
\frac{\tilde{\rho}_{j+1}}{\tilde{\rho}_j} = \frac{a_0^{(j+1)}}{b_0^{(j+1)}} \frac{b_0^{(j)}}{a_0^{(j)}} \frac{\rho_{j+1}}{\rho_j} = \frac{\omega_j \rho_{j+1}}{\alpha_j \rho_j}, \Rightarrow \beta_j = \frac{\alpha_j \tilde{\rho}_{j+1}}{\omega_j \tilde{\rho}_j}. \tag{1.15}
\]

We also need a formula to compute \( \alpha_j \), which can be obtained by the leading-term analysis:

\[
\alpha_j = \frac{(\phi_j(A)r_0) \cdot (\phi_j(A^t)r'_0)}{(A \pi_j(A)r_0) \cdot (\phi_j(A^t)r'_0)} = \frac{(\phi_j(A)r_0) \cdot (b_0^{(j)}(A^t)r'_0)}{(A \pi_j(A)r_0) \cdot (b_0^{(j)}(A^t)r'_0)}
\]

\[
= \frac{(\phi_j(A)r_0) \cdot (a_0^{(j)} r'_0)}{(A \pi_j(A)r_0) \cdot (a_0^{(j)} r'_0)} = \frac{(\phi_j(A)r_0) \cdot (\psi_j(A^t)r'_0)}{(A \pi_j(A)r_0) \cdot (\psi_j(A^t)r'_0)},
\]

thus:

\[
\alpha_j = \frac{r_j \cdot r'_0}{(Ap_j) \cdot r'_0} = \frac{\tilde{\rho}_j}{(Ap_j) \cdot r'_0}. \tag{1.16}
\]

2 The BICGSTAB Algorithm: Stabilization

Next we determine the value of \( \omega_j \). Let:

\[
s_j = r_j - \alpha_j Ap_j, \tag{2.1}
\]

then the next residual is given by:

\[
r_{j+1} = (I - \omega_j A)s_j. \tag{2.2}
\]

Since \( s_j \) is known at this point, an obvious choice for \( \omega_j \) is that who minimizes the \( L^2 \)-norm of \( r_{j+1} \), or equivalently:

\[
\omega_j = \frac{(As_j) \cdot s_j}{(As_j) \cdot (As_j)}. \tag{2.3}
\]
Finally, a formula is needed to compute the next approximation \( x_{j+1} \) which gives rise to the residual \( r_{j+1} \). Since:

\[
 r_{j+1} = s_j - \omega_j As_j = r_j - \alpha_j Ap_j - \omega_j As_j = b - Ax_j - \alpha_j Ap_j - \omega_j As_j,
\]

then clearly the solution is:

\[
 x_{j+1} = x_j + \alpha_j p_j + \omega_j s_j.
\]

(2.4)

Note that we no longer have a line search, c.f. the conjugate gradient method.

At last, we relate the BICGSTAB residuals to BCG residuals. Particularly, let \( r_j^B \) and \( r_j^G \) be the \( j \)-th residuals obtained by the two methods, respectively, we have:

\[
 r_j^B = \psi_j(A)r_j^G = (I - \omega_j A)\psi_{j-1}(A)r_j^G = \cdots = \prod_{k=0}^{j} (I - \omega_0 A)r_j^G.
\]

The first observation is that if \( r_j^G = 0 \), then \( r_j^B = 0 \), indicating the convergence of BICGSTAB as long as BCG converges to the true solution, providing exact arithmetics. Furthermore, the product generally creates a smoothing effect when we move from one iteration to the next, and for practical computations people usually observe smoother convergence (to zero) of BICGSTAB residuals comparing to that of BCG residuals. For example, Figure 1 compares the convergence of the three methods we’ve described so far for solving a two-dimensional diffusion equation [1].

![Figure 1: Convergence of BICGSTAB, BCG, and CGS for solving \(- (D_{u_x})_x - (D_{u_y})_y = 1\) by a finite difference method [1].](image)

Note that although BICGSTAB proves to be more robust than BCG in practice, it may still diverge when \( r_i \cdot r_0^* \) is close to zero (namely the “serious breakdown” of the Lanczos biorthogonalization procedure). In practice, people usually employ the restart strategy whenever this happens.

At the end of this lecture we list the full algorithms for a few of the CG-based methods we’ve discussed. Here \( \varepsilon_0 \) is a small tolerance that we choose in order to determine when to stop the iterations.

3
Algorithm 2.1 Conjugate Gradient (CG)
1: Compute $r_0 = b - Ax_0$, $p_0 = r_0$
2: for $j = 0, 1, \ldots$ do
3: $\alpha_j = (r_j \cdot r_j) / ((Ap_j) \cdot p_j)$
4: $x_{j+1} = x_j + \alpha_j p_j$
5: $r_{j+1} = r_j - \alpha_j Ap_j$
6: if $||r_{j+1}|| < \varepsilon_0$ then
7: Break;
8: end if
9: $\beta_j = (r_{j+1} \cdot r_{j+1}) / (r_j \cdot r_j)$
10: $p_{j+1} = r_{j+1} + \beta_j p_j$
11: end for
12: Set $x = x_{j+1}$

Algorithm 2.2 Conjugate Gradient Squared (CGS)
1: Compute $r_0 = b - Ax_0$, choose $r'_0$ such that $r_0 \cdot r'_0 \neq 0$
2: Set $p_0 = u_0 = r_0$
3: for $j = 0, 1, \ldots$ do
4: $\alpha_j = (r_j \cdot r'_0) / ((Ap_j) \cdot r'_0)$
5: $q_j = u_j - \alpha_j Ap_j$
6: $x_{j+1} = x_j + \alpha_j (u_j + q_j)$
7: $r_{j+1} = r_j - \alpha_j A(u_j + q_j)$
8: if $||r_{j+1}|| < \varepsilon_0$ then
9: Break;
10: end if
11: $\beta_j = (r_{j+1} \cdot r'_0) / (r_j \cdot r'_0)$
12: $u_{j+1} = r_{j+1} + \beta_j q_j$
13: $p_{j+1} = u_{j+1} + \beta_j (q_j + \beta_j p_j)$
14: end for
15: Set $x = x_{j+1}$
Algorithm 2.3 Biconjugate Gradient Stabilized (BICGSTAB)

1: Compute $r_0 = b - Ax_0$, choose $r'_0$ such that $r_0 \cdot r'_0 \neq 0$
2: Set $p_0 = r_0$
3: for $j = 0, 1, \ldots$ do
4: \quad $\alpha_j = (r_j \cdot r'_0) / ((Ap_j) \cdot r'_0)$
5: \quad $s_j = r_j - \alpha_j Ap_j$
6: \quad $\omega_j = ((As_j) \cdot s_j) / ((As_j) \cdot (As_j))$
7: \quad $x_{j+1} = x_j + \alpha_j p_j + \omega_j s_j$
8: \quad $r_{j+1} = s_j - \omega_j As_j$
9: \quad if $||r_{j+1}|| < \varepsilon_0$ then
10: \quad \quad Break;
11: \quad end if
12: \quad $\beta_j = (\alpha_j / \omega_j) \times (r_{j+1} \cdot r'_0) / (r_j \cdot r'_0)$
13: \quad $p_{j+1} = r_{j+1} + \beta_j (p_j - \omega_j Ap_j)$
14: end for
15: Set $x = x_{j+1}$

Numerical Exercise

In this exercise, we use the Conjugate Gradient (CG) method [2.1] the CGS algorithm [2.2] and the BICGSTAB algorithm [2.4] to solve several linear systems that stem from practical applications. Note that this BICGSTAB method is slightly different from the previous one in the following:

- After computing $s_j$, we check if it is close to zero. Indeed, as $s_j$ is given in residual form (with corresponding solution $x'_j = x_j + \alpha_j p_j$), we should terminate the algorithm if $x'_j$ gives a small enough residual.

- At the end of every loop, we check if “serious breakdown” will occur. The value $1e-6$ could be replaced by any other small positive number; but for this exercise we’ll just use the former.
Algorithm 2.4 Biconjugate Gradient Stabilized (BICGSTAB) with restart

1: Compute $r_0 = b - Ax_0$, choose $r'_0$ such that $r_0 \cdot r'_0 \neq 0$
2: Set $p_0 = r_0$
3: for $j = 0, 1, \ldots$ do
4: $\alpha_j = (r_j \cdot r'_0)/(Ap_j \cdot r'_0)$
5: $s_j = r_j - \alpha_jAp_j$
6: if $||s_j|| < \varepsilon_0$ then
7: $x_{j+1} = x_j + \alpha_jp_j$
8: Break;
9: end if
10: $\omega_j = (As_j \cdot s_j)/(As_j \cdot As_j)$
11: $x_{j+1} = x_j + \alpha_jp_j + \omega_js_j$
12: $r_{j+1} = s_j - \omega_jAs_j$
13: if $||r_{j+1}|| < \varepsilon_0$ then
14: Break;
15: end if
16: $\beta_j = (\alpha_j/\omega_j) \times (r_{j+1} \cdot r'_0)/(r_j \cdot r'_0)$
17: $p_{j+1} = r_{j+1} + \beta_j(p_j - \omega_jAp_j)$
18: if $|r_{j+1} \cdot r'_0| < 1e-6$ then
19: $r'_0 = r_{j+1}$
20: $p_{j+1} = r_{j+1}$
21: end if
22: end for
23: Set $x = x_{j+1}$

For all the algorithms, we set $\varepsilon_0 = 1e-6$. Note that most linear systems that are obtained directly from the physical problem are in general difficult to solve, whichever method we choose. In practice, we usually at least make use of some preconditioner to improve the convergence of the iterative methods. These matrices will be denoted by $M$ and further explained in each sub-problem below.

1) We first consider a symmetric positive definite matrix $A$ that results from discretizing a 2D heat equation. The matrix $A$ and the right hand side $b$ are provided in the files mat_sym_A.txt and mat_sym_b.txt, respectively. The format is described at the end of this exercise. For this problem, we use the incomplete Cholesky factorization to compute a preconditioner $M$ that is stored in mat_sym_M.txt.

The basic idea of the incomplete Cholesky factorization is to compute a lower-triangular matrix $L$ such that $LL^t \approx A$, and we set $M = L^{-1}$. Hence we expect $MAM^t \approx I$ to be better conditioned than $A$ itself. To this end, instead of solving $x$ for $Ax = b$, we solve $y$ such that $MAM^t y = Mb$ — that is, we apply the CG, CGS, and BICGSTAB to solve the system for $y$:

$$ (MAM^t)y = Mb, \quad (2.5) $$

rather than the original one for $x$.

2) We next consider a non-symmetric matrix $A$ that results from discretizing a 2D advection-diffusion equation. The matrix $A$, the right hand side $b$, and the precondition $M$ are stored in
mat_unsym_A.txt, mat_unsym_b.txt, and mat_unsym_M.txt, respectively. For this problem, the preconditioner \( M \) is obtained from the incomplete LU factorization. That is, we find a lower triangular matrix \( L \) and an upper triangular matrix \( U \) such that \( LU \approx A \), and compute \( M = U^{-1}L^{-1} \) so that \( MA \approx I \). Finally, instead of solving \( Ax = b \) we apply the CGS and BICGSTAB to solve the following equation:

\[
(MA)x = Mb. 
\]  

Objective: Implement the CG, CGS, and BICGSTAB as provided, and use the parameter \( \varepsilon_0 = 10^{-6} \) and initial guess \( x_0 = 0 \). Write a subroutine to read the matrix/vector information from the provided data files, which are organized as below:

- Each vector file contains \( n + 1 \) lines, where \( n \) is the dimension of the vector. The first line is an integer with the value \( n \), and the rest \( n \) lines correspond to the \( n \) components in the natural order.

- For each matrix file, the first line contains three integers in the form:

\[
n \quad n \quad N
\]

where \( n \) is the dimension of the matrix (i.e., an \( n \times n \) matrix), and \( N \) is the number of non-zero entries of the matrix. The rest of the file contains \( N \) lines, each line includes two integers and one real number:

\[
i \quad j \quad \alpha
\]

which means that the \( (i,j) \)-th component of the matrix has the value \( \alpha \). The numbers in the same line are separated by a single space.

Next, apply the CG, CGS, and BICGSTAB to solve the linear system for \( y \) (2.5); and apply the CGS and BICGSTAB to solve the linear system for \( x \) (2.6). Finally, plot the residual histories in a way similar to Figure 1.

References